Fraktale mit MuPAD, Eigene Lindenmayersysteme

Prof. Dr.Dörte Haftendorn, April 03, April 2013

```
Inhalt....: Erzeugung von Fraktalen mit Lindenmayersystemen eigne Version
Kategorie.: Arbeitsblatt
Mathematik: Fraktale Geometrie
MuPAD.....: 2.5.0
Datum.....: 2003-04-24  2013-04-03
Autoren...: Dörte Haftendorn <Haftendorn@uni.leuphana.de>
Funktionen: plot, plot::Lsys, plot::Turtle
```

LEVEL 1

Die Grundidee der Lindenmayersysteme ist folgende:

1. Es gibt gerade Striche, symolisiert durch F, "die Turle läuft **F**orward".
2. Es gibt gerade Vorwärtssprünge, symolisiert durch f, "die Turle springt **f**orward".
3. Es gibt einen Winkel w, um den die Turtle ihre Richtung ändern kann,
   + dreht w Grad nach links, - dreht w Grad nach rechts.
4. Mit [merkt sich die Turtle ihre momentane Position. Bei] kehrt sie dahin zurück.
5. Es gibt ein Startelement, ein "Axiom", einen Initiator, zum Beispiel einen geraden Strich F.
6. Es gibt eine Liste von "Regeln", nach denen die vorkommenden Zeichen erstetzt weden.
7. In einer "Generation"= Iterations-Stufe = Iterations-Schritt werden alle Regeln "auf einen Schlag" angewandt.
8. Das so entstehende "Lindenmayer-Wort" wird dann von der Turtle als Laufanweisung "abgearbeitet".

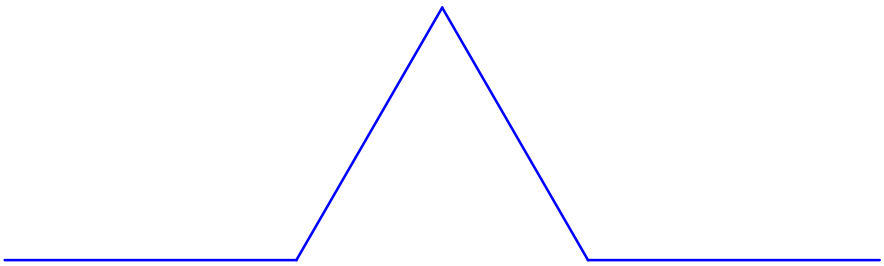
```
delete x,T,L:
F:=proc() begin T::forward(x) end_proc:
f:=proc() begin T::penUp; T::forward(x); T::penDown; end_
R:=proc() begin T::right(w)end_proc:
L:=proc() begin T::left(w)end_proc:
K:=proc() begin T::push() end_proc:
Z:=proc() begin T::pop() end_proc:
Rd:=proc() begin T::setLineColor([1,0,0]) end_proc:
Gr:=proc() begin T::setLineColor(RGB::Green) end_proc:
Bk:=proc() begin T::setLineColor(RGB::Black) end_proc:
Ma:=proc() begin T::setLineColor(RGB::Magenta)end_proc:
```

Kochkurve

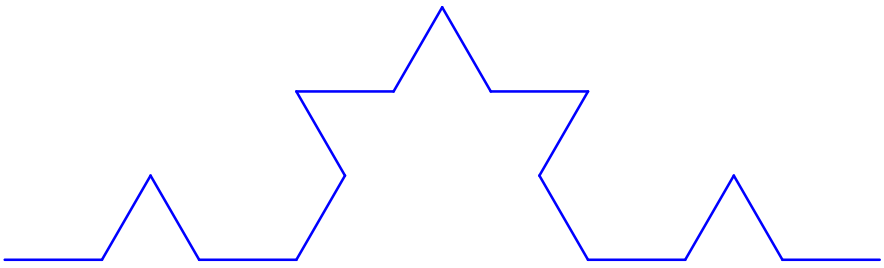
F( ) ist der Initiator.
Für die jeweils nächste Stufe wird jedes F( ) durch den Generator F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):
ersetzt. Erzeugung "von Hand":

```
x:=1:w:=PI/3:
T := plot::Turtle(LineWidth=2):    /*die Turtle wird erze
T::right(PI/2):  /* Startrichtung rechts */

T := plot::Turtle():  T::right(PI/2):
F():L():F():R():R():F():L():F():
plot(T, Axes = None,Scaling=Constrained):
```

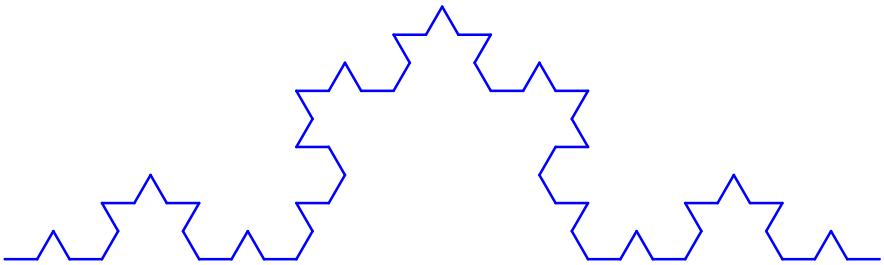


```
x:=1:w:=PI/3:T := plot::Turtle():  T::right(PI/2):
F():L():F():R():R():F():L():F():L():F():L():F():R():
R():F():L():F():R():R():F():L():F():R():R():F():L():F():I
plot(T, Axes = None,Scaling=Constrained):
```



```
x:=1:w:=PI/3:T := plot::Turtle():  T::right(PI/2):
F():L():F():R():R():F():L():F():L():F():L():F():R():R():F
L():F():L():F():R():R():F():L():F():L():F():L():F():R():F
R():R():F():L():F():R():R():F():L():F():L():F():L():F():F
L():F():L():F():R():R():F():L():F():L():F():L():F():R():F
```

```
plot(T, Axes = None,Scaling=Constrained):
```



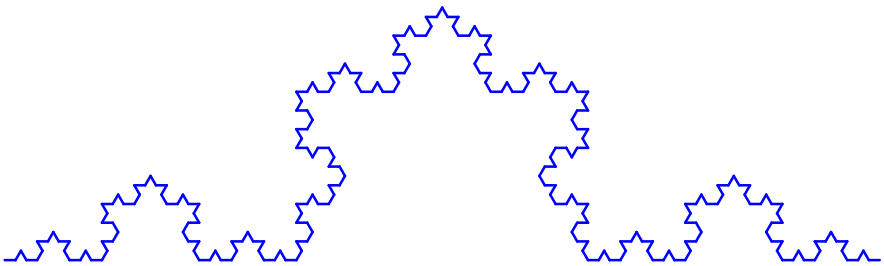
--F wird F-F++F-F , das wird dann zu F-F++F-F - F-F++F-F ++ F-F++F-F - F-F++F-F
da wird dann wieder jedes F durch die Regel erstetzt.

Da die Lindenmayerworte so schnell wachsen, ist hier noch etwas Programmierarbeit angesagt.

```
x:=1:w:=PI/3:T := plot::Turtle():  T::right(PI/2):
F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L(
):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L(
):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F( ):L( ):F(
):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F(
):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R(
):R( ):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F(
):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R(
):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F(
):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):
L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R(
):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F(
):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F( ):L(
):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L(
):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F(
):R( ):R( ):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F(
):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R(
):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F(
):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):
R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R(
):R( ):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F(
):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R(
```

```
):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F(
):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L(
):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L(
):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R(
):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F(
):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R(
):R( ):F( ):L( ):F( ):
L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R(
):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F(
):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F( ):L(
):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L(
):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F(
):R( ):R( ):F( ):L( ):F( ):R():R():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F(
):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R(
):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F( ):L( ):F( ):R():R():F(
):L( ):F( ):R( ):R( ):F( ):L( ):F( ):L():F( ):L( ):F( ):R( ):R( ):F(
):L( ):F( ):
```

```
plot(T, Axes = None,Scaling=Constrained):
```

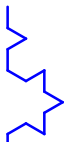


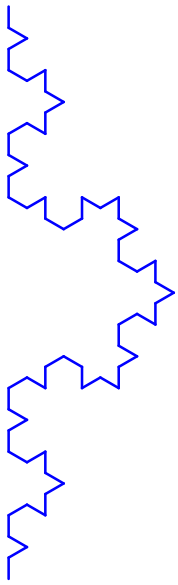
Eingebaute Lindenmyersysteme

Kochkurve

Stufe 1 zeigt immer den "Generator", die Figur die den Initiator ersetzt.

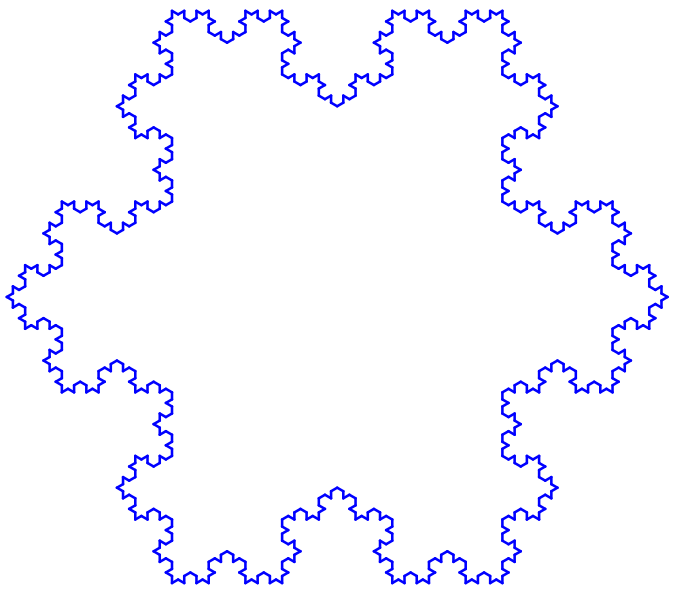
```
L := plot::Lsys(PI/3, "F", "F"="F-F++F-F",Generations=3):
plot(L, Axes = None,Scaling=Constrained)
```



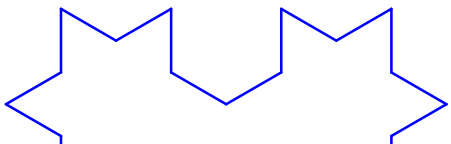
Koch Schneeflocke

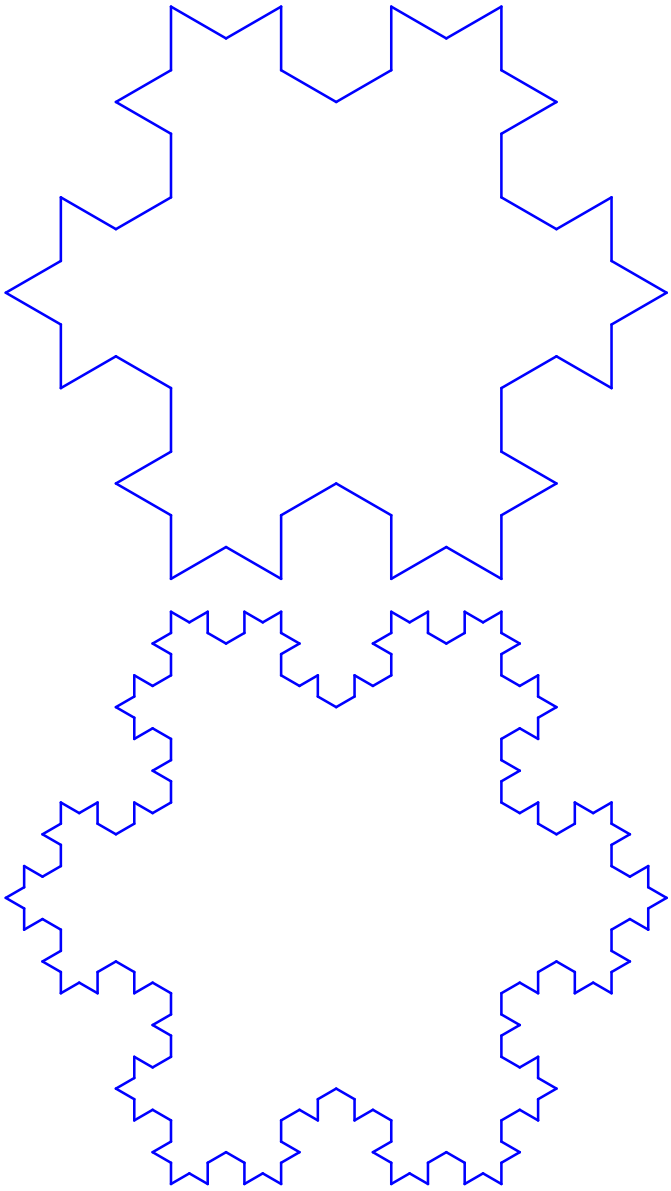
```
L := plot::Lsys(2*PI/6, "F++F++F", "F"="F-F++F-F"):
L::Generations := 4:
plot(L, Axes = None,Scaling=Constrained)
```



Gemeinsame Erzeugung mehrerer Generationen (Stufen)

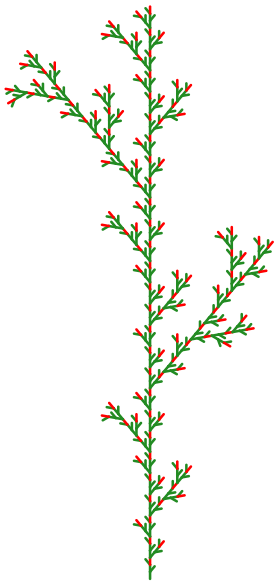
```
for g from 2 to 3 do
   L::Generations := g:
plot(L, Axes = None,Scaling=Constrained)
end_for:
```



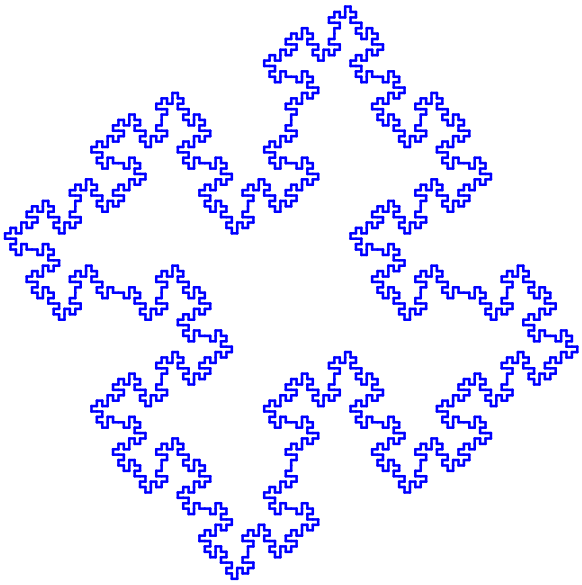
Zweig

```
delete(L):
L := plot::Lsys(2*PI/9,"GF","F"="GF[-F]F[+F]HF",
"G"=RGB::ForestGreen,"H"=RGB::Red):
L::Generations := 4:
plot(L, Axes = None,Scaling=Constrained)
```

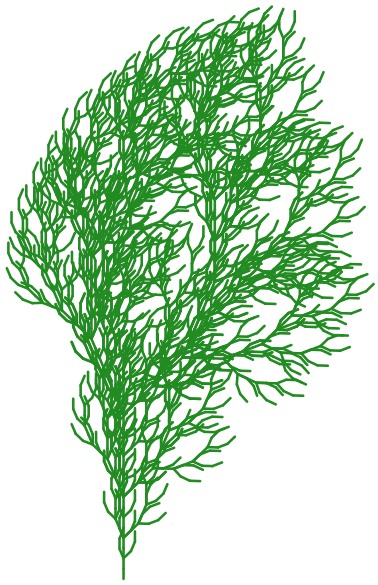
Quadro-Kochkurve

```
L := plot::Lsys(PI/2, "F-F-F-F",
"F"="F-F+F+FF-F-F+F"):
L::Generations := 3:
plot(L, Axes = None)
```

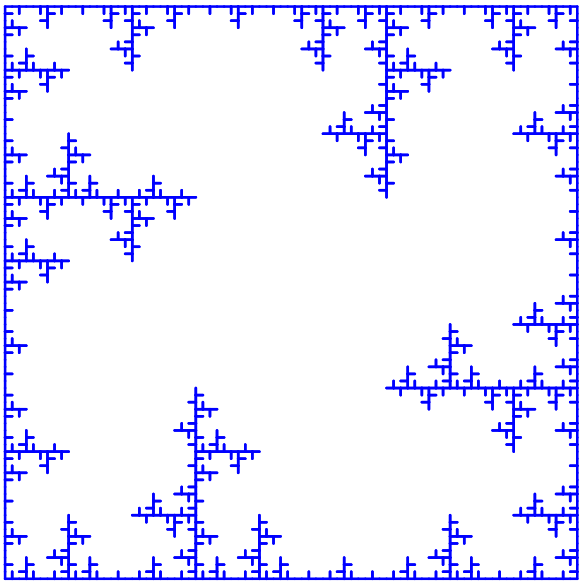


Wedel-Busch

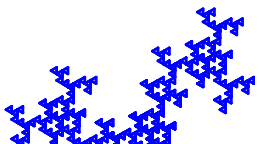
```
plot(plot::Lsys(23*PI/180, "GF", "F" = "FF-[-F+F+F]+[+F-F-F]",
"G"=RGB::ForestGreen,              Generations = 4))
```

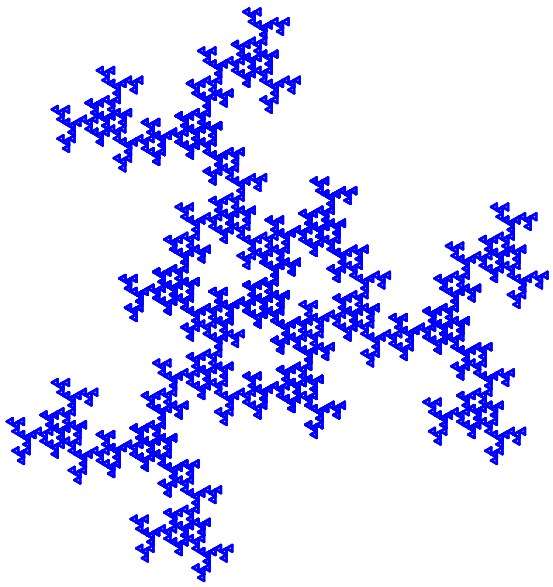



```
L := plot::Lsys(PI/2, "F-F-F-F-", "F"="FF-F--F-F"):
L::Generations := 4:
plot(L, Axes = None)
```

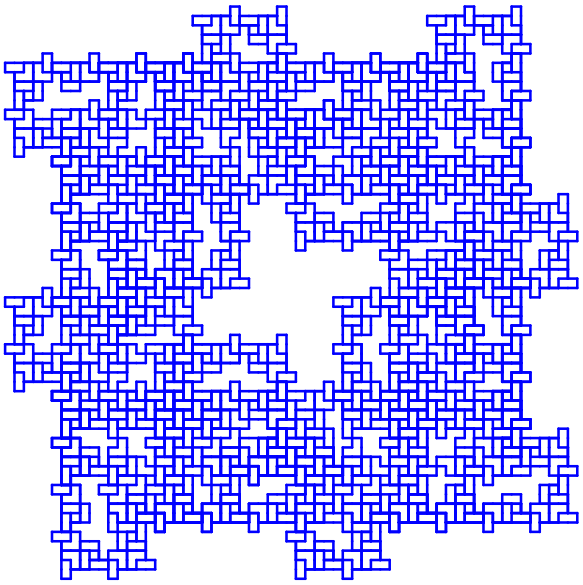


```
L := plot::Lsys(4*PI/3, "F-F-F-F-F-F-", "F"="FF-F--F-F"):
L::Generations := 5:
plot(L, Axes = None)
```

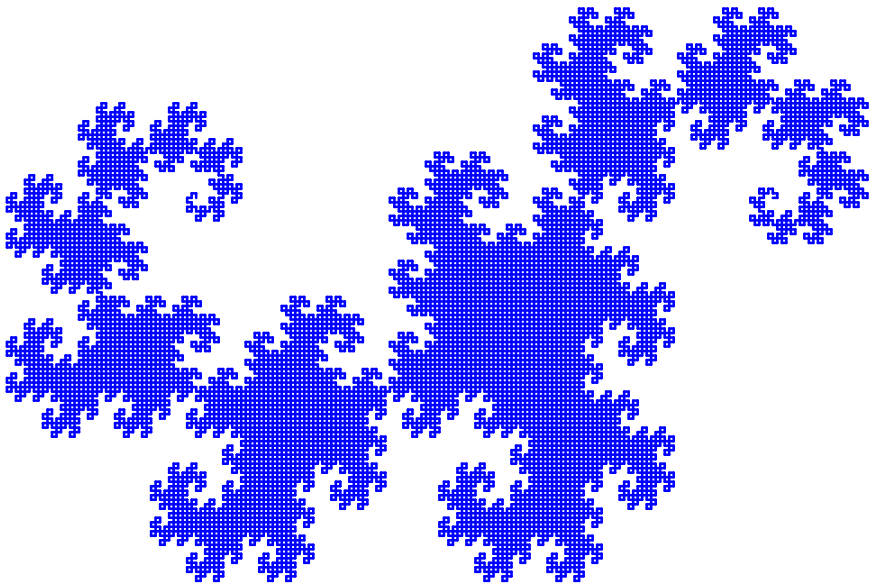


```
L := plot::Lsys(PI/2, "F-F-F-F", "F"="FF-F+F-F-FF"):
L::Generations := 4:
plot(L, Axes = None)
```

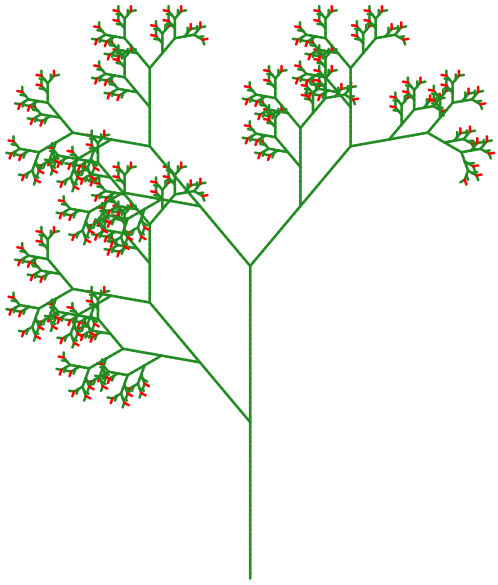


Drachenkurve

```
L := plot::Lsys(PI/2, "L", "L"="L+R+",
 "R"="-L-R", "L"=Line, "R"=Line):
L::Generations := 14:
plot(L, Axes = None)
```

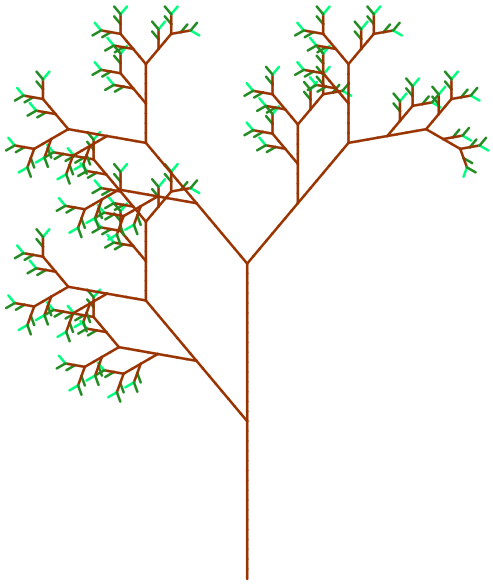



```
L := plot::Lsys(2*PI/9, "GL", "L"="GR[+L]R[-L]+HL",
"R"="RR", "L"=Line, "R"=Line,"G"=RGB::ForestGreen,"H"=RGB
L::Generations := 6:
plot(L, Axes = None)
```



Laubbaum

```
L := plot::Lsys(2*PI/9, "L", "L"="BR[+HL]BR[-GL]+HL", "R"
                "L"=Line, "R"=Line,Generations=a, a=1..5,
                "B"=RGB::Brown, "H"=RGB::ForestGreen,
                "G"=RGB::SpringGreen):
plot(L, Axes = None)
```

Achtung, animiert, hineinklicken!!!!

Koch Schneeflocke

```
L := plot::Lsys(2*PI/6, "F++F++F", "F"="F-F++F-F"):
L::Generations := 4:
plot(L, Axes = None,Scaling=Constrained)
```